


ANSIBLE BEST PRACTICES - WINDOWS



CLASS PAGE

<https://jrueles.github.io/ansible-windows-best>



© 2025 by Innovation In Software Corporation

ANSIBLE VAULT

© 2025 by Innovation In Software
Corporation



ANSIBLE VAULT



Ansible Vault encrypts variables and files so you can protect sensitive content such as passwords or keys rather than leaving it visible as plaintext in playbooks or roles.

To use Ansible Vault you need one or more passwords to encrypt and decrypt content.

Use the passwords with the `ansible-vault` command-line tool to create and view encrypted variables, create encrypted files, encrypt existing files, or edit, re-key, or decrypt files. You can then place encrypted content under source control and share it more safely.

© 2025 by Innovation In Software Corporation

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE VAULT



Ansible Vault can prompt for a password every time, or you can configure it to use a password file.

```
#ansible.cfg
[defaults]
    vault_password_file = ~/.vault_pass
```

© 2025 by Innovation In Software Corporation

5

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE VAULT



Each time you encrypt a variable or file with Ansible Vault, you must provide a password. When you use an encrypted variable or file in a command or playbook, you must provide the same password that was used to encrypt it.

© 2025 by Innovation In Software Corporation

6

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

POP QUIZ: DISCUSSION

Things to consider:

- Do you want to encrypt all your content with the same password, or use different passwords for different needs?
- Where do you want to store your password(s)?



ANSIBLE VAULT



Small teams can use a single password for everything encrypted. Store the vault password securely in a file or secret manager.

If you have a large team or many sensitive values to manage it is recommended to use multiple passwords.

You can use different passwords for different users or different levels of access. Depending on your needs, you might want a different password for each encrypted file, for each directory, or for each environment.


ANSIBLE VAULT



You might have a playbook that includes two vars files, one for the dev environment and one for the production environment, encrypted with two different passwords.

When you run the playbook, select the correct vault password for the environment you are targeting, using a vault ID.

VAULT ID



A vault ID is an identifier for one or more vault secrets.

Vault IDs provide labels to distinguish between individual vault passwords.

To use vault IDs, you must provide an ID label of your choosing and a source to obtain its password (either `prompt` or a file path):

```
--vault-id label@source
```

This switch is available for all commands that interact with vaults:


- `ansible-vault`
- `ansible-playbook`
- etc.

© 2025 by Innovation In Software Corporation

10

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE VAULT



Create a new encrypted data file

```
ansible-vault create foo.yml
```

Prompt for vault password

```
ansible-playbook --ask-vault-pass myplay.yml
```

Use password file

```
ansible-playbook --vault-password-file pass myfile.yml
```

© 2025 by Innovation In Software Corporation

11

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE VAULT



Common vault commands

```
# Edit encrypted files
ansible-vault edit playbook.yml

# Rekeying
ansible-vault rekey play.yml task.yml report.yml

# Encrypt existing files
ansible-vault encrypt foo.yml bar.yml baz.yml

# Decrypting files
ansible-vault decrypt task.yml run.yml play.yml

# View encrypted files
ansible-vault view break.yml fix.yml fun.yml
```

Should you wish to change your password on a vault-encrypted file or files, you can do so with the rekey command:



TAGS



If you have a large playbook, it may become useful to be able to run only a specific part of it rather than running everything in the playbook. Ansible supports a `tags` attribute for this reason.

Tags can be applied at multiple levels including:

- Tasks
- Roles
- Plays
- Blocks

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

TAGS



If you have a large playbook, it may become useful to be able to run only a specific part of it rather than running everything in the playbook. Ansible supports a `tags` attribute for this reason.

Tags can be applied at multiple levels including:

- Tasks
- Roles
- Plays
- Blocks

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

TAGS



At the simplest level, you can apply one or more tags to an individual task. You can add tags to tasks in playbooks, in task files, or within a role.

It is also possible to add the same tag to multiple tasks.

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

TAGS



Here's an example showing tasks to install and configure software. Using tags, it is possible to specify which task runs.

```
tasks:
- name: Install packages
  win_chocolatey:
    name:
      - procexp
      - putty
      - windirstat
    state: present
  tags:
    - packages
    - chocolatey
- name: Configure
  win_emplate:
    src: templates/src.j2
    dest: "C:\Windows\System32\file.conf"
  tags:
    - configuration
```

© 2025 by Innovation In Software Corporation

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

TAGS



This example shows the 'dev' tag assigned to multiple tasks.

```
- name: Install notepad++  
  win_chocolatey:  
    name: notepadplusplus  
    state: present  
    tags: dev  
  
- name: Install curl  
  win_chocolatey:  
    name: curl  
    state: present
```

If you ran these tasks in a playbook with `--tags ntp`, Ansible would run the one tagged `ntp` and skip the other.

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

TAGS



Inheritance:

No one wants to add the same tag to multiple tasks. To avoid repeating code you can tag the play, block, or role. Ansible applies the tags down the dependency chain to all child tasks.

Blocks are useful for applying a tag to many, but not all, of the tasks in your play.

Plays are better suited if every task in the play should have the same tags.

Adding tags to roles allows you to run specific roles.

TAGS



Define tags at the block level

```
- name: sql express
  tags: sql
  block:
    - name: Install sql
      win_chocolatey:
        name: sql-server-express
        state: present

    - name: Enable and run sql
      win_service:
        name: 'MSSQL${ sql_instance_name }'
        state: started
        enabled: yes

    - name: Install utils
      win_chocolatey:
        name: procexp
        state: present
      tags: utils
```

© 2025 by Innovation In Software Corporation

20

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

TAGS



Ansible reserves two tags:

- Always
 - Run the task/play unless specifically skipped
 - `--skip-tags always`
- Never
 - Skip unless specifically requested
 - `--tags never`

POP QUIZ: DISCUSSION

What are tags used for?





ANSIBLE COLLECTIONS



You can extend Ansible by adding custom modules or plugins. You can create them from scratch or copy existing ones for local use.

A simple way to share plugins and modules with your team or organization is by including them in a collection and publishing the collection on Ansible Galaxy.

ANSIBLE COLLECTIONS



Modules:

Modules are reusable, standalone scripts that can be used by the Ansible API, the `ansible` command, or the `ansible-playbook` command.

Modules provide a defined interface. Each module accepts arguments and returns information to Ansible by printing a JSON string to `stdout` before exiting. Modules execute on the target system (usually that means on a remote system) in separate processes.

Plugins:

Plugins extend Ansible's core functionality and execute on the control node within the `/usr/bin/ansible` process. Plugins offer options and extensions for the core features of Ansible - transforming data, logging output, connecting to inventory, and more

ANSIBLE GALAXY



Use ansible-galaxy to install collection

```
ansible-galaxy (collection/role) install
```

You can install from the community, or any .tar.gz file.

```
ansible-galaxy collection install azure.azurecollection
```

Use new collection (full namespace, collection, module)

```
- name: Azure collection
  hosts: localhost
  tasks:
    - azure.azurecollection.azure_rm_storageaccount:
        resource_group: myRG
        name: myStorageAccount
        account_type: Standard_LRS
```

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE GALAXY



Use ansible-galaxy to install collection or role

```
ansible-galaxy (collection/role) install
```

You can install from the community, or any .tar.gz file.

```
ansible-galaxy collection install azure.azcollection
```

Use new collection (full namespace, collection, collections element)

```
- name: Azure collection
hosts: localhost
collections:
  - azure.azcollection
tasks:
  - azure_rm_storageaccount:
      resource_group: myRG
      name: myStorageAccount
      account_type: Standard_LRS
```

© 2025 by Innovation In Software Corporation

27

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE GALAXY



Use ansible-galaxy to install role

```
ansible-galaxy role install
```

You can install from the community, or any .tar.gz file.

```
ansible-galaxy collection install azure.azure_modules
```

Use new role:

```
- name: Azure role
  hosts: localhost
  roles:
    - { role: azure.azure_modules }
  tasks:
    - name: create storage account
      azure_rm_storageaccount:
        resource_group: MyRG
        name: MyStorage
        account_type: Standard_LRS
```

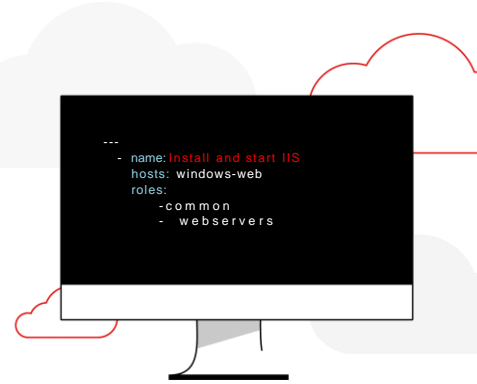
© 2025 by Innovation In Software Corporation

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.

ANSIBLE ROLES

What are they?

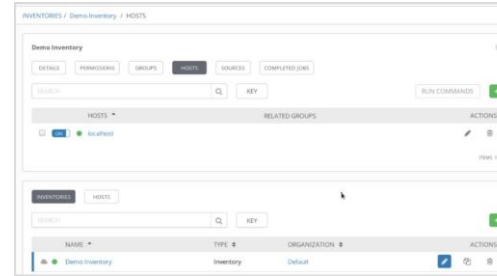
Group your tasks and variables of your automation in a reusable structure. Write roles once and share them with others who have similar challenges in front of them.





ANSIBLE INVENTORY

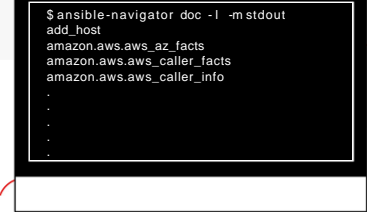
- Manage inventory with Automation Platform
- This is a static inventory with no host groups and one host, called `localhost`.
- Clicking on that host in the inventory reveals that the inventory variable `ansible_connection: local` is set.



ACCESSING ANSIBLE DOCS

With the use of the latest command utility `ansible-navigator`, one can trigger access to all the modules available to them as well as details on specific modules.

A formal introduction to `ansible-navigator` and how it can be used to run playbooks in the following exercise.



```
$ ansible-navigator doc -l -m stdout
add_host
amazon.aws.aws_az_facts
amazon.aws.aws_caller_facts
amazon.aws.aws_caller_info
.
```


ACCESSING ANSIBLE DOCS

Aside from listing a full list of all the modules, you can use `ansible-navigator` to provide details about a specific module.

In this example, we are getting information about the `user` module.

```
$ ansible-navigator doc user -m stdout  
  
> ANSIBLE.BUILTIN.USER  
(/usr/lib/python3.8/site-packages/ansible/modules/user.py)  
  
Manage user accounts and user attributes.  
For Windows targets, use the  
[ansible.windows.win_user] module  
instead.
```

ANSIBLE PLAYBOOK VARIABLES



```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }}" "{{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```

ANSIBLE PLAYBOOK VARIABLES



Ansible playbooks

```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }}" "{{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{ var_three }}"
```

ansible is awesome

ANSIBLE FACTS

- ▶ Just like variables, really...
- ▶ ...but: coming from the host itself!
- ▶ Check them out with the setup module



ANSIBLE FACTS



Ansible playbooks

```
---  
- name: facts playbook  
  hosts: localhost  
  
  tasks:  
    - name: Collect all facts of host  
      setup:  
        gather_subset:  
          - 'all'
```

```
$ ansible-navigator run playbook.yml
```

ANSIBLE NAVIGATOR TUI

Ansible Navigator TUI

PLAY NAME	OK	CHANGED	UNREACHABLE	FAILED	SKIPPED	IGNORED	IN PROGRESS	TASK COUNT	PROGRESS
0 facts playbook	D2	0	0	0	0	0	0	2	COMPLETE

RESULT	HOST	NUMBER	CHANGED	TASK	TASK ACTION	DURATION
0 OK	localhost	0	False	Gathering Facts	gather_facts	1s
1 OK	localhost	1	False	Collect all facts of host	setup	1s

PLAY [facts playbook:1]

TASK [Collect all facts of host]

OK: [localhost]

.

12 | ansible_facts:

13 | ansible_all_ipv4_addresses:

14 | - 10.0.2.100

15 | ansible_all_ipv6_addresses:


16 | - fe80::1caa:f0ff:fe15:23c4

© 2025 by Innovation In Software Corporation

38

CONDITIONALS VIA VARS

Example of using a variable labeled *my_mood* and using it as a conditional on a particular task.



```
vars:
  my_mood: happy

tasks:
  - name: task, based on my_mood var
    debug:
      msg: "Yay! I am({ my_mood})!"
    when: my_mood == "happy"
```

19

© 2025 by Innovation In Software Corporation

39

CONDITIONALS VIA VARS



Ansible Conditionals

```
---  
- name: variable playbook test  
  hosts: localhost  
  
  vars:  
    my_mood: happy  
  
  tasks:  
    - name: task, based on my_mood var  
      debug:  
        msg: "Yay! I am {{ my_mood }}!"  
      when: my_mood == "happy"
```

Alternatively

```
- name: task, based on my_mood var  
  debug:  
    msg: "Ask at your own risk. I'm {{ my_mood }}!"  
  when: my_mood == "grumpy"
```


CONDITIONALS WITH FACTS



Ansible Conditionals w/ Facts

```
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: install apache
    apt:
      name: apache2
      state: latest
      when: ansible_distribution == 'Debian' or
            ansible_distribution == 'Ubuntu'

  - name: install IIS
    win_feature:
      name: Web-server
      state: latest
      when: ansible_distribution == 'Windows'
```

41

TASK STATE



Using Previous Task State

```
---
- name: variable playbook test
  hosts: localhost

  tasks:
    - name: Ensure iis is present
      win_feature:
        name: Web-server
        state: latest
        register: iis_results

    - name: Restart iis
      service:
        name: W3Svc
        state: restart
      when: iis_results.changed
```

42

VARIABLES AND LOOPS



Ansible Variables & Loops

```
---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
    - name: Ensure user is present
      win_user:
        name: dev_user
        state: present

    - name: Ensure user is present
      win_user:
        name: qa_user
        state: present

    - name: Ensure user is present
      win_user:
        name: prod_user
        state: present
```

VARIABLES AND LOOPS



Ansible Variables & Loops

```
---  
- name: Ensure users  
  hosts: node1  
  become: yes  
  
  tasks:  
    - name: Ensure user is present  
      win_user:  
        name: "{{item}}"   
        state: present  
      loop:  
        - dev_user  
        - qa_user  
        - prod_user
```

RUNNING A PLAYBOOK

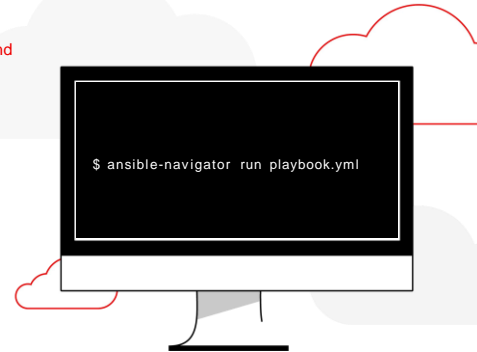
Using the latest ansible-navigator command



What is ansible-navigator?

ansible-navigator command line utility and text-based user interface (TUI) for running and developing Ansible automation content.

It replaces the previous command used to run playbooks "ansible-playbook".



ANSIBLE NAVIGATOR

Bye ansible-playbook, Hello ansible-navigator

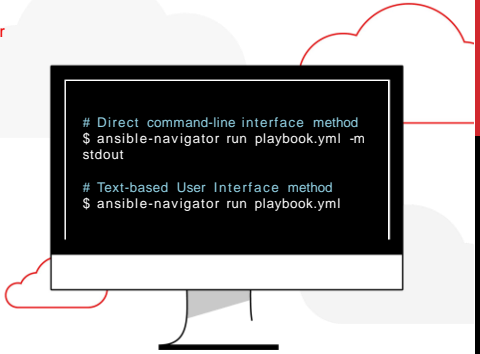


How do I use ansible-navigator?

As previously mentioned, it replaces the ansible-playbook command.

As such it brings two methods of running playbooks:

- ▶ Direct command-line interface
- ▶ Text-based User Interface (TUI)



```
# Direct command-line interface method
$ ansible-navigator run playbook.yml -m
stdout

# Text-based User Interface method
$ ansible-navigator run playbook.yml
```

ANSIBLE NAVIGATOR

Mapping to previous Ansible commands

ansible command	ansible-navigator command
ansible-config	ansible-navigator config
ansible-doc	ansible-navigator doc
ansible-inventory	ansible-navigator inventory
ansible-playbook	ansible-navigator run

ANSIBLE NAVIGATOR

Common subcommands

Name	Description	CLI Example	Colon command within TUI
collections	Explore available collections	ansible-navigator collections --help	:collections
config	Explore the current ansible configuration	ansible-navigator config --help	:config
doc	Review documentation for a module or plugin	ansible-navigator doc --help	:doc
images	Explore execution environment images	ansible-navigator images --help	:images
inventory	Explore and inventory	ansible-navigator inventory --help	:inventory
replay	Explore a previous run using a playbook artifact	ansible-navigator replay --help	:replay
run	Run a playbook	ansible-navigator run --help	:run
welcome	Start at the welcome page	ansible-navigator welcome --help	:welcome

AUTOMATION CONTROLLER

© 2025 by Innovation In Software
Corporation



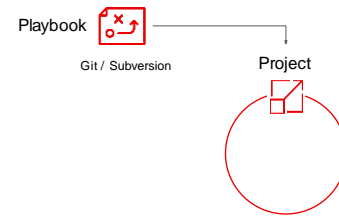
What is Ansible Automation Controller?

Ansible Automation Controller is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

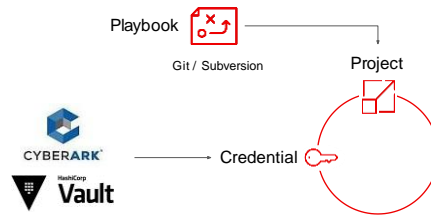
- ▶ Role-based access control
- ▶ Deploy entire applications with push-button deployment access
- ▶ All automations are centrally logged
- ▶ Powerful workflows match your IT processes



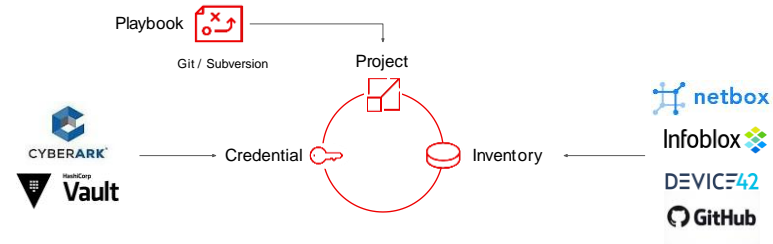
ANATOMY OF AN AUTOMATION JOB



ANATOMY OF AN AUTOMATION JOB



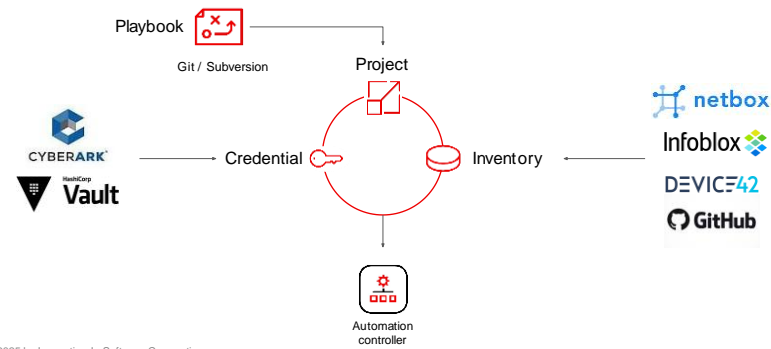
ANATOMY OF AN AUTOMATION JOB



© 2025 by Innovation In Software Corporation

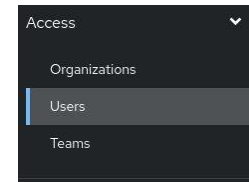
53

ANATOMY OF AN AUTOMATION JOB



USER MANAGEMENT

- An **organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization.
- A **user** is an account to access Ansible Automation Controller and its services given the permissions granted to it.
- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.



© 2025 by Innovation In Software Corporation

56

ORGANIZATION

- Newly created users inherit specific roles from their organization based on their user type.
- Assign additional roles to a user after creation to grant permissions to view, use, or change other Ansible Platform objects.

Organizations

<input type="checkbox"/>	Name	<input type="text"/>	<input type="button" value="Add"/>	<input type="button" value="Delete"/>	1 - 2 of 2	<	>
	Name	1	Members	Teams	Actions		
<input type="checkbox"/>	Default		0	0			<input type="button" value="Edit"/>
<input type="checkbox"/>	NewOrg		0	0			<input type="button" value="Edit"/>
					1 - 2 of 2 items	<	>
					1	of 1 page	>

TEAM

- You can apply permissions at the team level.

Teams

Create New Team

Name * MyNewTeam

Description

Organization * NewOrg

Save Cancel

USER ROLES

- An organization is also one of these objects.
- There are three roles that users can be assigned:
- Admin
- Auditor
- User

Users

Create New User

First Name: User, Last Name: One, Email: user1@domain.com

Username: user1, Password: [masked], Confirm Password: [masked]

User Type: ☒ Normal User, ☐ System Auditor, ☐ System Administrator

Organization: NewOrg

Save Cancel

ROLES

- Create custom roles with specific permissions.

Add team permissions

1 Add resource type

2 Select items from list

3 Select roles to apply

Job templates

Workflow job templates

Credentials

Inventories

Projects

Organizations

Next

Back

Cancel

INVENTORY

- Log into Ansible Platform (the admin user will work for this example).
- Click on **Inventories**.
- In the INVENTORIES window, click the + button.
- Enter a NAME for the inventory and its ORGANIZATION (often "Default")



The screenshot shows the 'NEW INVENTORY' form in the Ansible Platform. The form has a sidebar on the left with navigation links: Home, Dashboard, Jobs, Overview, My team, Inventory, Hosts, Credentials, Groups, and Organizations. The main form area has tabs for 'GENERAL', 'ADVANCED', 'HOSTS', 'CREDENTIALS', and 'GROUPS'. The 'GENERAL' tab is active. It contains fields for 'NAME' (with a dropdown menu), 'DESCRIPTION' (text area), 'ORGANIZATION' (dropdown menu), 'HOSTS' (text area), and 'CREDENTIALS' (dropdown menu). There are 'CANCEL' and 'CREATE' buttons at the bottom right.

INVENTORY

- In the Ansible Platform GUI, click the **Inventories** menu, then click on the name of the inventory.
 - Click the **HOSTS** button, then click on **+**. This displays the “Create a new host” tooltip.
 - In the **HOST NAME** field enter the hostname or IP address of the managed host.
 - In the **VARIABLES** text box, you can set values for variables that apply only to this host.
 - Click **SAVE**.

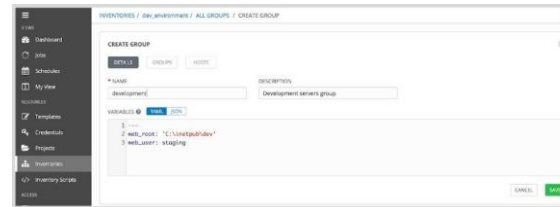


INVENTORY GROUPS

- Groups allow you to organize hosts into a set that can be managed together
- Hosts may be in multiple groups at the same time
 - All hosts that are in a particular data center
 - All hosts that have a particular purpose
 - Dev / Test / Prod hosts can be grouped
- Groups can be nested
 - The *europa* group might include a *paris_dc* group and a *london_dc* group
- This allows you to run playbooks on particular groups
- This allows you to set a variable to a specific value for all hosts in a group

INVENTORY GROUPS

- In the Ansible Platform GUI, click the **Inventories** menu, and click on the inventory to edit.
- Click the **GROUPS** button, then click on **+**. This will open the "Create a new group" tooltip.
- In the NAME field, enter the name of the group.
- Define any values for variables
- Click **SAVE**.



© 2025 by Innovation In Software Corporation

INVENTORY GROUPS

- In the Ansible Platform GUI, click the **Inventories** menu, and click on the inventory to edit.
- Click the **GROUPS** button, then click on the group to edit.
- Click the **HOSTS** button, then click on **+**. This will open the “Add a host” tooltip. Select “New Host”.
- In the HOST NAME field, enter the hostname or IP address of the managed host to add.
- Define any values for variables that affect only that host (overriding any group variables).
- Click **SAVE**.

INVENTORY ROLES

Role	Description
Admin	The inventory Admin role grants users full permissions over an inventory. These permissions include deletion and modification of the inventory. In addition, this role also grants permissions associated with the inventory roles Use , Ad Hoc , and Update .
Use	The inventory Use role grants users the ability to use an inventory in a job template resource. This controls which inventory is used to launch jobs using the job template's playbook.
Ad Hoc	The inventory Ad Hoc role grants users the ability to use the inventory to execute ad hoc commands.
Update	The inventory Update role grants users the ability to update a dynamic inventory from its external data source.
Read	The inventory Read role grants users the ability to view the contents of an inventory.

INVENTORY VARIABLES

When you manage a static inventory in the Ansible Platform web UI, you may define inventory variables directly in the inventory objects.

- Variables set in the inventory details affect all hosts in the inventory.
- Variables set in a group's details are the equivalent of `group_vars`.
- Variables set in a host's details are the equivalent of `host_vars`.

INVENTORY VARIABLES

INVENTORIES / MAIL SERVICES

MAIL SERVICES

DETAILS

PERMISSIONS

GROUPS

HOSTS

SOURCES

COMPLETED JOBS

* NAME

MAIL SERVICES

DESCRIPTION

* ORGANIZATION

Q Default

INSIGHTS CREDENTIAL

Q

INSTANCE GROUPS ⓘ

Q

VARIABLES ⓘ

TAB

JSON

EXPAND

1

...

1

CANCEL

SAVE

INVENTORY GROUP VARIABLES

INVENTORIES / MAIL SERVERS / ALL GROUPS / southeast

southeast

DETAILS GROUPS HOSTS

* NAME DESCRIPTION

southeast

VARIABLES [YAML](#) [JSON](#)

```
1 ---
2 ntp: ntp-se.example.com
```

CANCEL SAVE

INVENTORY HOST VARIABLES

INVENTORIES / MAIL SERVERS / HOSTS / boston1.example.com

boston1.example.com [On](#)

[DETAILS](#) [FACTS](#) [GROUPS](#) [COMPLETED JOBS](#)

* HOST NAME DESCRIPTION

VARIABLES [YAML](#) [JSON](#) [EXPAND](#)

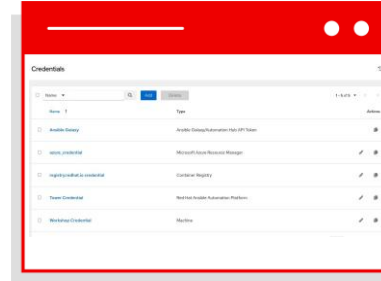
1 ---

CREDENTIALS

Credentials are utilized by Automation Controller for authentication with various external resources:

- ▶ Connecting to remote machines to run jobs
- ▶ Syncing with inventory sources
- ▶ Importing project content from version control systems
- ▶ Connecting to and managing network devices

Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.



CREDENTIALS

- Create credentials in the UI
- This credential contains information that is used to access managed hosts in the **Inventory**.

The screenshot shows a web interface for editing a credential. The title bar reads 'CREDENTIALS / EDIT CREDENTIAL'. The main content area is titled 'Demo Credential' and has two tabs: 'DETAILS' (active) and 'PERMISSIONS'. The form includes the following fields:

- * NAME**: A text input field containing 'Demo Credential'.
- DESCRIPTION**: A text input field.
- ORGANIZATION**: A dropdown menu with a search icon and the text 'SELECT AN ORGANIZATION'.
- * CREDENTIAL TYPE**: A dropdown menu with 'Machine' selected.
- TYPE DETAILS**: A section containing:
 - USERNAME**: A text input field with 'admin'.
 - PASSWORD**: A text input field with a search icon and a 'Prompt on launch' checkbox.
 - SSH PRIVATE KEY**: A text area with a hint: 'HINT: Drag and drop private file on the field below.'



ROLLING UPDATES



Batch size:

By default, Ansible will try to manage all the machines referenced in a play in parallel. For a rolling update use case, you can define how many hosts Ansible should manage at a single time by using the `serial` keyword:

ROLLING UPDATES



Example playbook utilizing `serial` keyword.

```
- name: test play
  hosts: webservers
  serial: 2
  gather_facts: False

  tasks:
    - name: task one
      command: hostname
    - name: task two
      command: hostname
```

With 4 hosts in the group 'webservers', 2 would complete the play before moving onto the next 2 hosts.

ROLLING UPDATES

In the previous example, if we had 4 hosts in the group 'webservers', 2 would complete the play before moving on to the next 2 hosts:

```
PLAY [webservers] *****
TASK [task one] *****
changed: [web2]
changed: [web1]

TASK [task two] *****
changed: [web1]
changed: [web2]

PLAY [webservers] *****
```

ROLLING UPDATES

Now that web1 & web2 are complete Ansible continues with web3 & web4

```
PLAY [webservers] *****
TASK [task one] *****
changed: [web3]
changed: [web4]

TASK [task two] *****
changed: [web3]
changed: [web4]

PLAY RECAP *****
web1 : ok=2 changed=2 unreachable=0 failed=0
web2 : ok=2 changed=2 unreachable=0 failed=0
web3 : ok=2 changed=2 unreachable=0 failed=0
web4 : ok=2 changed=2 unreachable=0 failed=0
```

ROLLING UPDATES



The `serial` keyword can also be specified as a percentage, which will be applied to the total number of hosts in a play, in order to determine the number of hosts per pass:

```
- name: test play
  hosts: webservers
  serial: 30%
```

If the number of hosts does not divide equally into the number of passes, the final pass will contain the remainder.

ROLLING UPDATES



The batch size can be specified as a list with integer, or percent:

```
- name: test play
hosts: webservers
serial:
  - 1
  - 5
  - 10
```

Above the first batch would contain a single host, next 5, and if any left, each would have 10 until complete.

ROLLING UPDATES



The batch size can be specified as a list with integer, or percent:

```
- name: test play
hosts: webservers
serial:
  - "10%"
  - "20%"
  - "100%"
```

ROLLING UPDATES



Maximum Threshold Percentage:
Ansible executes tasks on all hosts in the defined group unless `serial` is defined.

In some situations, such as with the rolling updates, it may be desirable to abort the play when a certain threshold of failures have been reached. To achieve this, you can set a maximum failure percentage on a play as follows:

```
- hosts: webservers
  max_fail_percentage: 30
  serial: 10
```

In the above example, if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted.